TCS — Blockchain Tutorial 2

# Explore Blockchain with a Simple JavaScript Example

To reinforce your understanding of the role of blockchain we illustrate the main concepts by building a simple blockchain using JavaScript. This tutorial is adapted from a YouTube video that very nicely illustrates how the blockchain works:

1. Part 1 — [Create a Blockchain in JavaScript](https://www.youtube.com/watch?v=zVqczFZr124)

* This part of the tutorial implements the basic blockchain structure in JavaScript.

1. Part 2 — [Implementing a ‘Proof of Work’ in JavaScript](https://www.youtube.com/watch?v=HneatE69814)

* This part of the tutorial adds a proof of work method to increase the difficulty of calculating the validity of the block

1. Part 3 — Refine your JavaScript blockchain

* This part of the tutorial is optional exercise asking you to convert the JavaScript blockchain to utilise the SQLIte database. You will need:
  + A new SQLIte PDO database connection
  + PHP PDO model classes and associated methods to reflect the JavaScript Classes and methods
  + PHP controller methods to reflect the JavaScript methods in Part 1 and 2
  + Use JavaScript and JQuery to represent the miner process and invoke the new controller methods to append validated blocks to the new blockchain (based on SQLite)
  + Process any returned messages via a JSON object, e.g. false block, true block, etc.
  + …

You may simply click on the links above to follow the tutorials in YouTube or follow the brief written tutorials Part 1 and Part 2 below, Part 3 is down to you to investigate.

Let’s get started with Part 1.

## Part 1 — Create a Blockchain in JavaScript

The initial presentation gave you a feel for what a blockchain is, how it is structured, how it works. Part 1 of this Blockchain Tutorial 2 is based on the YouTube video indicated about, and has been selected because it very neatly illustrates the important blockchain concepts, while also allowing you to reinforce these concepts through code experimentation.

However, we should be aware that several underlying concepts are missing. So, this blockchain would not work very well in practice.

In this Blockchain Tutorial 2 Part 2, we will call this blockchain 'myBitcoin' and we will use OOP for JavaScript. You will recall we used OOP for PHP to create our Model in our Blockchain Tutorial 2.

So, step by step:

1. First, we should create a JavaScript file in our MVC frameworks scripts folder. Call this JavaScript file jsBlockchain.js. Add, a comment of some sort to the top, good practice.
2. Next, let’s create a block class. We start by creating a constructor class called 'block' to represent a block in the blockchain. This block class will hold methods to represent the main components of the block, such as: the block hash, previous hash, data, timestamp, and so on, Figure 1.



Figure : Create the Class Block and its constructor.

* Code explanation
  + Line 2-9: This is the Block class with its constructor
  + Line 5-7: This is the constructor of the class Block that holds the block properties: index, timestamp, data, and previoushash. From line 6 we will add some methods.
    - 'index' is where the block sits in the blockchain.
    - 'timestamp' tells us when the block was created.
    - 'data' holds any type of data you want to hold in the block, e.g. bitcoins transacted, address of sender, address of receiver, etc.
    - 'previoushash' is the hash value of the previous block in the blockchain, which insures the integrity of the blockchain, and is initially set to empty, as we have no previoushash in the genesis block.

1. We now need to keep track of the block properties: index, timestamp, data and previoushash using the JavaScript keyword this to own the Block class constructor object properties (I think), Figure 2.

* Code explanation:
  + 7 – 10: The value of ‘this’ in JavaScript, will become the new object when the constructor is used to create an object. Check this out: [Gentle explanation of ‘this’ keyword in JavaScript](https://dmitripavlutin.com/gentle-explanation-of-this-in-javascript/). Also, read this [[quir]{mode]](https://www.quirksmode.org/js/this.html) for a good explanation.
    - The upshot of this is that when you invoke the construction (with its class Block properties) this.index = index; this.timestamp = timestamp, etc. become global variables (I think).
  + Line 12: We also have to calculate the current block hash, so we will need a global variable for this, which at the moment is set to empty

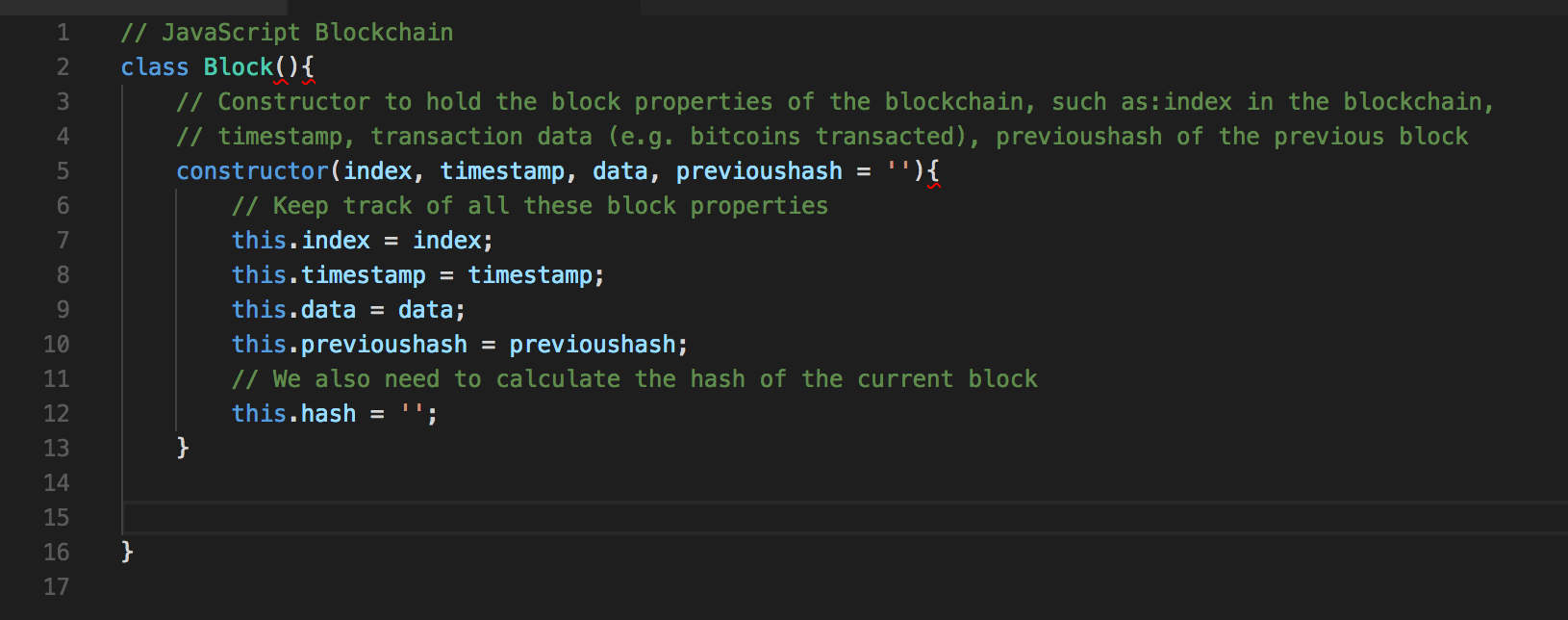


Figure : Keep track of the block properties

1. We need to create a method to calculate the current block hash using a JavaScript hash library such as SHA256. This method in the constructor will take the current block properties index, timestamp, data, previoushash and run them through a hash function to create the current block hash used to identfy this block on the blockchain.   
     
   Now, JavaScript does not have a native hash library so we need to import a hash library, e.g. crypto.js. I am using Visual Studio Code for my developer environment, because it has lots of nice features such as: an integrated terminal window, integrated GitHub, and so on. So, you will need to access a terminal window to install crypto.js.   
     
   When you have opened up a terminal window you should check that you have node.js installed, which is an open source server framework. There are many online resources that show you how to install node.js. In my case, I just checked it was already install by typing node -v in my terminal window, which returned v6.10.0, which is the version I have already installed.   
     
   This means that NPM, the node package manage, is already installed, so we can use this to install crypto.js. Cool!  
     
   You will need to type ‘npm install --save crypto-js’ on the terminal command line. You will also need to decide where to install the SHA256 library and as these are JavaScript libraries, perhaps you should change directory in your terminal window to your scripts folder and install there  
     
   Go ahead and type ‘npm install --save crypto-js’ on the terminal command line. This will install the node\_modules folder containing the crypto-js folder contacting in turn a load of JavaScript files including the sha256.js file needed to calculate the current block hash. Figure 3 shows the installation on the terminal window in Visual Studio Code.



Figure : Install the node\_modules/crypto-js/sha256.js files used to calculate the current block hash value

1. Now we can start to create the calculate hash method and use the SHA256 library to do this, Figure 4.

* Code explanation
  + Line 2: This now includes (requires) the SHA256 JavaScript hash library, which is used to calculate the hash value of the current block.
  + Line 19 – 23: This is the calculateHash() method in the Block class, which uses the SHA256 JavaScript hash library to calculate the hash. We input to the SHA256 function the current index, previoushash, timestamp and data. In doing so we convert the JavaScript data into JSON string format, and also cast the hash result into a string format, otherwise SHA256 returns an object.
  + Line 15: We now modify this.hash to calculate the current block hash value.

## ../../../../Desktop/Screen%20Shot%202017-11-23%20at%2015.37.05.png

Figure 4: Add the calculatehash() method

1. Next, we should create a blockchain class with a constructor that initialises the blockchain.

## Part 2 — Implementing a ‘Proof of Work’ in JavaScript

…